XML-based Phrase Alignment in Parallel Treebanks

Martin Volk, Sofia Gustafson-Capková, Joakim Lundborg,
Torsten Marek, Yvonne Samuelsson, Frida Tidström
Stockholm University
Department of Linguistics
106 91 Stockholm, Sweden
volk@ling.su.se

Abstract

This paper describes the usage of XML for representing cross-language phrase alignments in parallel treebanks. We have developed a TreeAligner as a tool for interactively inserting and correcting such alignments as an independent level of treebank annotation.

1 Introduction

The combined research on treebanks and parallel corpora has recently led to parallel treebanks. A parallel treebank consists of syntactically annotated sentences in two or more languages, taken from translated (i.e. parallel) documents. In addition, the syntax trees of two corresponding sentences are aligned on a sub-sentential level. This means word level, phrase level and clause level, but we will refer to it as phrase alignment since it best represents the idea. Parallel treebanks can be used as training or evaluation corpora for word and phrase alignment, as input for example-based machine translation (EBMT), as training corpora for transfer rules, or for translation studies.

We are developing an English-German-Swedish parallel treebank. In this paper we will focus on the representation of the treebank and the alignment. We will briefly explain the steps for building the parallel treebank and describe our new alignment tool. This paper is a follow-up and revision of (Samuelsson and Volk, 2005) based on fresh insights from this tool.

2 Building the treebanks

Our parallel treebank contains the first two chapters of Jostein Gaarder's novel "Sofie's World" with about 500 sentences.[1] In addition it contains 500 sentences from economy texts (a quarterly report by a multinational company as well as part of a bank's annual report).

In creating the parallel treebank, we have first annotated the monolingual treebanks with the ANNOTATE treebank editor.[2] It includes Thorsten Brants' statistical Part-of-Speech Tagger and Chunker. The chunker follows the TIGER annotation guidelines for German (Brants and Hansen, 2002), which gives a flat phrase structure tree. This means, for instance, no unary nodes, no "unnecessary" NPs (noun phrases) within PPs (prepositional phrases) and no finite VPs (verb phrases).

Using a flat tree structure for manual treebank annotation has two advantages for the human annotator: fewer annotation decisions, and a better overview of the trees. This comes at the prize of the trees not being complete from a linguistic point of view. Moreover, flat syntax trees are also problematic for node alignment in a parallel treebank. We prefer to have "deep trees" to be able to draw the alignment on as many levels as possible; in fact, the more detailed the sentence structure is, the more expressive our alignment can become.

As an example, let us look at the work flow for the German-Swedish parallel treebank. We first annotated the German sentences semi-automatically in the flat manner, and we then automatically deepened the flat syntax trees (Samuelsson and Volk, 2004).

[1] A prototype of the parallel treebank was developed by Yvonne Samuelsson and contains the first chapter of the novel in German and Swedish. Later, a French version was added and aligned to the Swedish treebank by (Tidström, 2005). We would like to thank Eckhard Bick, Declan Groves and Jörg Tiedemann for their help.

[2] www.coli.uni-sb.de/sfb378/negra-corpus/annotate.html

We annotated the Swedish sentences by first tagging them with a Part-of-Speech tagger trained on SUC (the Stockholm-Umeå Corpus). Since we did not have a Swedish treebank to train a Swedish chunker, we used a trick to apply the German chunker for Swedish sentences. We mapped the Swedish Part-of-Speech tags in the Swedish sentences to the corresponding German tags. Since the German chunker works on these tags, it then suggested constituents for the Swedish sentences, assuming they were German sentences. These experiments and the resulting time gain were reported in (Volk and Samuelsson, 2004). Upon completion of the Swedish treebank with flat syntax trees, we applied the same deepening method as for German, and we then converted the Part-of-Speech labels back to the Swedish labels.

Finally, we annotated the English sentences according to the Penn Treebank guidelines. We trained the PoS tagger and the chunker on the Penn Treebank and integrated them into ANNOTATE. The English guidelines lead to complete trees so that the deepening step is not needed.

3 XML Representation of the Trees

After finishing the monolingual treebanks with ANNOTATE, the trees were exported from the accompanying SQL database and converted into TIGER-XML. TIGER-XML is a line-based (i.e. not nested and thus database-friendly) representation for graph structures, which includes syntax trees with node labels, edge labels, multiple features on the word level and even crossing edges.[3]

In a TIGER-XML graph each leaf (= token) and each node (= linguistic constituent) has a unique identifier which is prefixed with the sentence number. Leaves are numbered from 1 to 499 and nodes starting from 500 (under the plausible assumption that no sentence will ever have more than 499 tokens). As can be seen in the following example, node 500 in sentence 12 is of the category PP (prepositional phrase). The phrase consists of word number 4, which is the preposition *in*, plus node 502 which in turn is marked as an NP (noun phrase), consisting of the words 5 and 6. It should be noted that the `id` attribute in the token lines serves a dual purpose of identifier and order marker. This makes it possible to represent crossing branches.

```
<s id="s12">
<graph root="s12_501">
<terminals>
 <t id="s12_1" word="Jetzt" pos="ADV" />
 <t id="s12_2" word="bog" pos="VVFIN" />
 <t id="s12_3" word="sie" pos="PPER" />
 <t id="s12_4" word="in" pos="APPR" />
 <t id="s12_5" word="den" pos="ART" />
 <t id="s12_6" word="Kløverveien" pos="NE"/>
 <t id="s12_7" word="ein" pos="PTKVZ" />
 <t id="s12_8" word="." pos="$." />
</terminals>
<nonterminals>
 <nt id="s12_500" cat="PP">
  <edge label="HD" idref="s12_4" />
  <edge label="NK" idref="s12_502" />
 </nt>
<nt id="s12_502" cat="NP">
  <edge label="NK" idref="s12_5" />
  <edge label="HD" idref="s12_6" />
 </nt>
[...]
</nonterminals>
</graph>
</s>
```

This means that the token identifiers and constituent identifiers are used as pointers to represent the nested tree structure. This example thus represents the upper tree in figure 1.

One might wonder why tree nesting is not directly mapped into XML nesting. But the requirement that the representation format must support crossing edges rules out this option. TIGER-XML is a powerful representation format and is typically used with constituent symbols on the nodes and functional information on the edge labels. This constitutes a combination of constituent structure and dependency structure information.

4 XML Representation of the Alignment

Phrase alignment can be regarded as an additional layer of information on top of the syntax structure. We use the unique node identifiers for the phrase alignment across parallel trees. We also use an XML representation for storing the alignment. The alignment file first stores the names of the treebank files and assigns identifiers to them. Every single phrase alignment is then stored with the tag `align`. Thus the entry in the following example represents the alignment of node 505 in sentence 13 of language one (German) to the node 506 in sentence 14 of language two (Swedish).

```
<treebanks>
  <tbank file="Sofie_DE.xml" id="De"/>
  <tbank file="Sofie_SV.xml" id="Sv"/>
</treebanks>
<align type="exact">
  <node node_id="s13_505" tbank_id="De"/>
  <node node_id="s14_506" tbank_id="Sv"/>
</align>
```

[3] See www.ims.uni-stuttgart.de/projekte/TIGER

This representation allows phrase alignments within m:n sentence alignments, which we have used in our project. The XML also allows m:n phrase alignments, which we however have not used for reasons of simplicity and clarity. Two nodes are aligned if the words which they span convey the same meaning and could serve as translation units.

The alignment format allows alignments to be specified between an arbitrary number of nodes, for example nodes from three languages. And it includes an attribute `type` which we currently use to distinguish between exact and approximate alignments.

5 Our Tree Alignment Tool

After finishing the monolingual trees we want to align them on the phrase level. For this purpose we have developed a "TreeAligner". This program is a graphical user interface to insert (or correct) alignments between pairs of syntax trees.[4] The TreeAligner can be seen in the line of tools such as I*Link (Ahrenberg et al., 2002) or Cairo (Smith and Jahr, 2000) but it is especially tailored to visualize and align full syntax trees.

The TreeAligner requires three input files. One TIGER-XML file with the trees from language one, another TIGER-XML file with the trees from language two, plus the alignment file as described above. The alignment file might initially be empty when we want to start manual alignment from scratch, or it might contain automatically computed alignments for correction. The TreeAligner displays tree pairs with the trees in mirror orientation (one top-up and one top-down). See figure 1 for an example. This has the advantage that the alignment lines cross fewer parts of the lower tree. The trees are displayed with node labels and greyed-out edge labels. The PoS labels are omitted in the display since they are not relevant for the task.

Each alignment is displayed as a dotted line between one node (or word) from each tree. Clicking on a node (or a word) in one tree and dragging the mouse pointer to a node (or a word) in the other tree inserts an alignment line. Figure 2 shows an example of a tree pair with alignment lines. Currently the TreeAligner supports two types of alignment lines (displayed in different colors) which are used to indicate exact translation correspondence vs. approximate translation correspondence. However, our experiments indicate that eventually more alignment types will be needed to precisely represent different translation deviations.

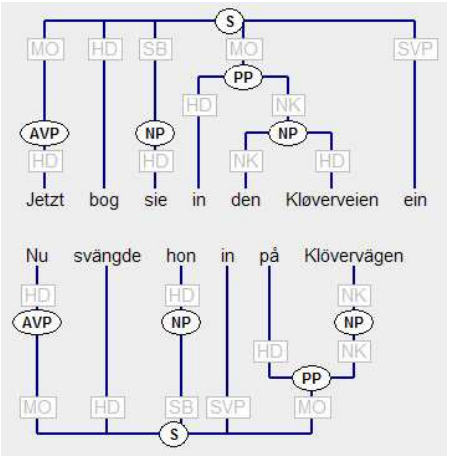

Figure 1: Tree pair German-Swedish in the TreeAligner.

Often one tree needs to be aligned to two trees in the other language. We therefore provide the option to scroll the trees independently. For instance, if we have aligned only a part of tree 20 from language one to tree 18 of language two, we may scroll to tree 19 of language two in order to align the remaining parts of tree 20.[5]

The TreeAligner is designed as a stand-alone tool (i.e. it is not prepared for collaborative annotation). It stores every alignment in an XML file (in the format described above) as soon as the user moves to a new tree pair. It has been tested on parallel treebanks with several hundred trees each.

6 Conclusion

We have shown a straightforward way to tie in XML-based phrase alignment information with syntax trees represented in TIGER-XML. The alignment information is stored independently from the treebank files. This independence allows for a modularization and separation of the annotation but it entails that the synchronization of the

[4]The TreeAligner has been implemented in Python by Joakim Lundborg and is freely available at www.ling.su.se/DaLi/downloads/treealigner/index.htm

[5]The final result of an m:n tree alignment can be visualized with an SVG-based display which we have described in (Samuelsson and Volk, 2005). SVG (Scalable Vector Graphics) describes vector graphics in XML.

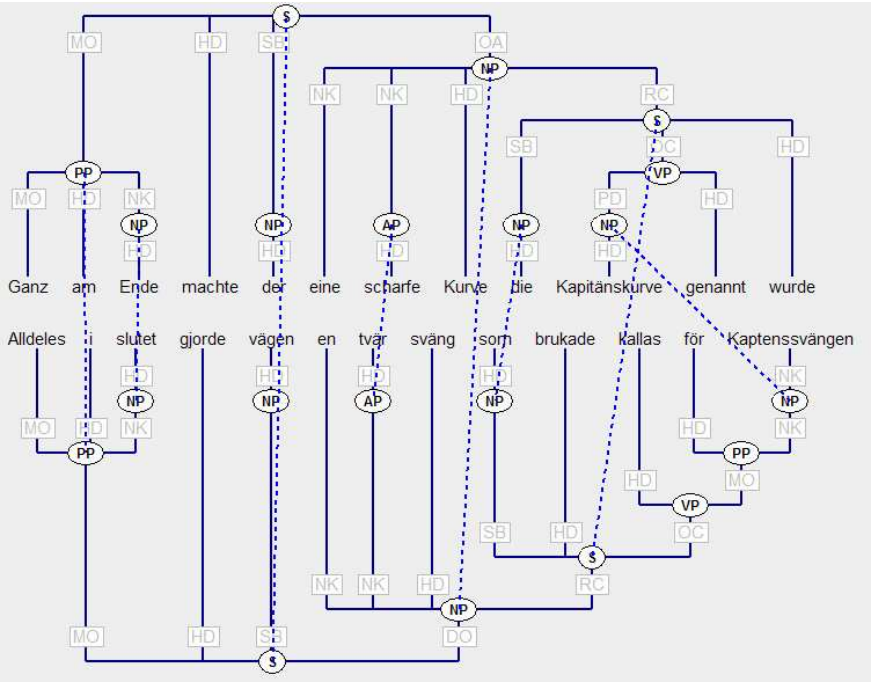

Figure 2: Tree pair German-Swedish with alignment in the TreeAligner.

treebanks with the alignment needs to be guarded separately. If any of the treebanks is modified, the modification of the alignment needs to follow.

We have argued for the use of a graphical TreeAligner to display and interactively modify the alignment between parallel syntax trees. The TreeAligner allows for m:n sentence alignment, word alignment and node alignment. And it supports the distinction between exact and approximate alignments.

As a next step we plan to integrate a component for automatic phrase alignment into the TreeAligner. The user can then select a tree pair and will get automatic phrase alignment predictions. We have already experimented with the projection of automatically computed word alignments to predict phrase alignment. Of course, the automatic phrase alignment has to be manually checked if we want to ensure high quality alignment data.

Another avenue of further research is the inclusion of yet more levels of annotation. For example, we are currently experimenting with the annotation of semantic frames on top of the treebanks. We use the SALSA tool developed at Saarbrücken University (Erk and Pado, 2004) which also assumes TIGER-XML input. So, TIGER-XML has become the lingua franca of treebank annotation which allows for the addition of arbitrary layers.

References

Lars Ahrenberg, Magnus Merkel, and Mikael Andersson. 2002. A system for incremental and interactive word linking. In *Proc. of LREC-2002*, pages 485–490, Las Palmas.

Sabine Brants and Silvia Hansen. 2002. Developments in the TIGER annotation scheme and their realization in the corpus. In *Proc. of LREC-2002*, pages 1643–1649, Las Palmas.

Katrin Erk and Sebastian Pado. 2004. A powerful and versatile XML format for representing role-semantic annotation. In *Proc. of LREC-2004*, Lisbon.

Yvonne Samuelsson and Martin Volk. 2004. Automatic node insertion for treebank deepening. In *Proc. of 3rd Workshop on Treebanks and Linguistic Theories*, Tübingen, December.

Yvonne Samuelsson and Martin Volk. 2005. Presentation and representation of parallel treebanks. In *Proc. of the Treebank-Workshop at Nodalida*, Joensuu, May.

Noah A. Smith and Michael E. Jahr. 2000. Cairo: An alignment visualization tool. In *Proc. of LREC-2000*, Athens.

Frida Tidström. 2005. Extending a parallel treebank with data in French. C-uppsats, Department of Linguistics, Stockholm University, April.

Martin Volk and Yvonne Samuelsson. 2004. Bootstrapping parallel treebanks. In *Proc. of Workshop on Linguistically Interpreted Corpora (LINC) at COLING*, Geneva.